Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

Claims 1-21 (Cancelled)

22. (Currently amended) A computer-implemented method comprising:

in parallel hardware threads executing in a processor comprising a plurality of microengines, receiving a source code line to be break pointed in a selected microengine;

determining whether the source code line can be break pointed;

if the source code line can be break pointed, identifying the selected microengine to insert a break point into, which microengine threads to enable breakpoints for, and which microengines to stop if a break point occurs; and

if the source code line cannot be break pointed, signaling an error.

23. (Currently amended) The computer-implemented method of claim 22 wherein identifying further comprises generating a break point routine by modifying a template of instructions stored in a debug library.

24. (Currently amended) The computer-implemented method of claim 23 wherein identifying further comprises inserting a break point at the source code line and a branch to the source code line.

25. (Currently amended) The computer-implemented method of claim 24 further comprising:

executing the parallel hardware threads until the break point is encountered; and

executing the break point routine, the break point routine stopping selected threads and determining which microengine sent an interrupt.

26. (Currently amended) The computer-implemented method of claim 25 further comprising displaying program information to a user.

27. (Currently amended) The computer-implemented method of claim 26 further comprising resuming execution of the parallel threads in response to a user input.

28. (Previously presented) A processor that can execute multiple parallel threads in multiple microengines and that comprises:

a register stack;

a program counter for each executing context;

an arithmetic logic unit coupled to the register stack and a program control store that stores a breakpoint routine that causes the processor to:

receive a source code line to be break pointed in a selected microengine;

determine whether the source code line can be break pointed;

if the source code line can be break pointed, identify the selected microengine to insert a break point into, which microengine threads to enable breakpoints for, and which microengines to stop if a break point occurs; and

if the source code line cannot be break pointed, signal an error.

29. (Previously presented) The processor of claim 28 wherein identifying further comprises generating a break point routine by modifying a template of instructions stored in a debug library.

30. (Previously presented) The processor of claim 29 wherein identifying further comprises inserting a break point at the source code line and a branch to the source code line.

31. (Previously presented) The processor of claim 30 wherein the breakpoint routine further causes the processor to:

execute the parallel threads until the break point is encountered; and

stop selected threads and determine which microengine sent an interrupt.

32. (Previously presented) The processor of claim 31 wherein the breakpoint routine further causes the processor to display program information to a user.

33. (Previously presented) The processor of claim 32 the breakpoint routine that causes the processor to resume execution of the parallel threads in response to a user input.